UNIX — A Short Overview

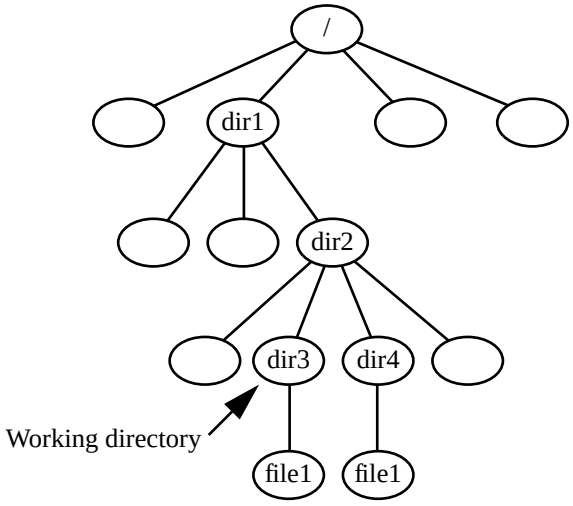
Composed by Christian Jacob, Fall 2000

1 Important terms in a UNIX environment

File: A file in UNIX is a sequence of characters. Each file has a name.

Catalog / directory: A catalog or directory is a special form of a file. It serves as a container, a folder, which contains other files, directories or links to other directories. Each directory contains a link to itself (“.”) and a link to its parent directory (“..”).

Filesystem: The UNIX filesystem, the set of all files, is structured like an upside-down tree:



The **root** node of the tree is a directory named “/”. Each directory node has as many branches as there are files in this directory.

Working directory and **pathnames**: The directory in which the user is currently working is denoted as the *working directory*. The pathname of your current working directory is issued by the command `pwd`. You change to your default working directory by using the command `cd`.

Usually, filenames are entered *relative* to your working directory.

Example 1: In the filesystem graphic above, `file1` refers to file `file1` in directory `dir3`, as `dir3` is the current working directory. In order to access `file1` in directory `dir4`, we have to “climb up” one level to `dir2` and then descend to directory `dir4`. Hence, the relative pathname would be:

```
../dir4/file1
```


We can also use *absolute* pathnames to access files. Absolute pathnames start at the root directory ("/") of the file system, where the directory names are separated by a slash ("/"), and end with the name of the file.

Example 2: The two files named `file1` in directories `dir3` and `dir4` have the absolute pathnames

```
/dir1/dir2/dir3/file1 and

/dir1/dir2/dir4/file1
```

Shell: A shell is a command-line interface that allows you to execute UNIX commands and application programs.

Commands: A command is an instruction issued to the UNIX operating system. Commands consist of a *name* and a sequence of *options* and *arguments*. Command names, options, and arguments are separated by space or by tab characters. Example: `ls -l dir3`.

Option: Options are most often single characters, preceeded by a "-" (minus). Options help to specify commands more precisely. Example: `ls -a`.

Argument: Arguments are usually names of files that are input to a command. Example: `cp file1 file1copy`.

Process: Any command or program that is currently executed is a process. You can get a list of the running processes on your machine by using the command `ps`.

Job: If a process is suspended by `Control-z` (press the Control- and z-key) the process becomes a (suspended) background job. Jobs can be controlled by job commands (see below).

&: A program or command with the "&" character appended is executed as a *background* job. This means that the user does not have to wait until the job has ended, but can continue working in the shell.

Input-output redirection: Processes perform their input and output via socalled *data streams*, the input and the output stream. By default, the input stream gets its data from the keyboard. Data sent to the output stream are displayed on th screen by default. However, these default stream settings can be changed by the commands "<", ">" and "|".

<: If the input stream for a command should receive its data from a file instead of the keyboard, one has to append "<" and the filename. The command `cat`, for example, reads characters from the keyboard and displays them on the screen, until the EOF (end of file) character is read (`Control-d`). The command

```
cat < file1
```

however, uses the data in file `file1` as input. Hence it will display the contents of file `file1`.

>: The output of a command can be redirected into a file by using the command ">". The command


```
ls > file2
```

for example, generates a file `file2` which contains all the filenames of the working directory. No output appears on the screen.

Pipes: The output stream of a command can be directly connected to the input stream of another command by using the *pipelining* command "|". For example, the output of the command

```
ls | wc -w
```

displays the number of files in the current working directory.

2 UNIX commands

This is an overview of the most important UNIX commands. Keep in mind that this is only a "starter kit" to get you introduced to UNIX.

Each command has to be completed by the RETURN key.

The **boldtype** names are the actual UNIX commands, all *italics* prints are either names or numbers that have to be entered appropriately.

2.1 General commands

date	Outputs date and time.
man *name*	Displays a description of the command *name* and its options. (Hit the spacebar for line-by-line output. Stop the output with "q".)
whoami	Returns the user name.
who	Gives a list of all users logged into your machine.
wc *name*	(*wordcount*) Counts the lines, words, and characters in file *name*.
passwd	Allows you to change your secret password.
exit	Exit the current shell.

2.2 File commands

ls	Lists all filenames in the working directory, which do not begin with a ".".
ls -a	Lists all filenames in the working directory.
ls -l	Lists all filenames in the working directory, with more information about the files, such as size, creation date, filetype, etc.
ls *name*	Lists all files in directory *name*.
cat *name*	Outputs the contents of the file *name*.
less *name*	Outputs the contents of the file *name* in a page-by-page fashion.
lpr -P*pri name*	(*line printer*) The file *name* is sent to the line printer *pri*.

mv *old new* (*move*) Change the filename from *old* to *new.*

rm *name* (*remove*) The file *name* is deleted.

cp *name name2* (*copy*) Create a copy of file *name*. The copied file has the name *name2*.

2.3 Catalog / directory commands

pwd (*print working directory*) Outputs the name of the working directory.

mkdir *name* (*make directory*) Create a new subdirectory with name *name*.

rmdir *name* (*remove directory*) The subdirectory *name* is removed if it does not contain any files.

cd *name* (*change directory*) The directory *name* becomes the new working directory.

cd .. The parent directory of the current working directory becomes the new working directory.

2.4 Job commands

jobs Issues a list of all jobs running for the user.each job has a unique number (displayed in square brackets).

fg *%n* Job number *n* becomes a foreground job.

bg *%n* Continue job number *n* as a background job.

stop *%n* Stop the job with number *n*.

kill *%n* Terminate the job with number *n* immediately.

2.5 Process commands

ps (*process status*) Lists all user processes.

kill *n* Process number *n* is stopped in a controlled fashion.

kill -9 *n* process number *n* is stopped immediately.

2.6 Important special characters and control sequences

Control-c The current program/command is stopped.

Control-z The current program/command is sent to the background and suspended.

Control-d End of file (EOF)

Control-s Stops output on the screen.

Control-q Continues output on the screen.

name **&** Execute the command *name* in the background.

